Kódování a komprese dat – Projekt – Komprese obrazových dat s využitím Huffmanova kódování

Dominik Nejedlý

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole
xnejed09@vutbr.cz

28. dubna 2024

1 Úvod

Tento projekt se zabývá vytvořením konzolové aplikace pro kompresi šedo tónových obrazových dat s uplatněním principu Huffmanova kódování. Aplikace se vstupními daty pracuje po jednotlivých bytech (hodnoty 0-255) a podporuje sekvenční i adaptivní skenování obrazu a rovněž využití modelu a kódování délkou sledů (RLE) pro předzpracování dat. Informace pro vypracování byly čerpány z přednášek předmětu Kódování a komprese dat (KKO).

2 Rozbor řešení

Implementačně je program rozdělen do několika částí. Hlavními jsou kódování délkou sledů, zkráceně RLE, (modul `rle`), model (modul `model`), Huffmanovo kódování (modul `huffman`) a skenování obrazu se samotnou kompresí (modul `compress`). Zbývající části (moduly `args` a `io`) zajišťují zpracování vstupních argumentů programu a načítání vstupních dat.

2.1 Kódování délkou sledů (RLE)

Posloupnosti stejných symbolů jsou nahrazeny trojicí symbolů (bytů), přičemž prvním z nich je specifikovaná RLE značka (výchozí RLE značkou v implementaci je hodnota 128), druhým je byte nesoucí počet opakování symbolu (počet opakování je 1-256 kódovaný vždy o 1 menší hodnotou, jelikož sled délky nula se nikdy nevyskytne), který je specifikován třetím symbolem (hodnota 0-255). Jelikož jsou sledy nahrazovány 3 symboly, jsou kódovány pouze sledy délky 4 a více. RLE značka je vždy kódována jako sled. Jelikož jsou pak sledy všech ostatních symbolů delší než 3, lze sledy symbolu RLE značky kratší než 4 jednoznačně kódovat pouze s využitím 2 symbolů (bytů). Zakódovaný sled pomocí RLE je proto delší než nezakódovaný sled pouze pro samostatný výskyt RLE značky, tedy pouze v případě sledu symbolu RLE značky délky 1.

Pokud je RLE využito, probíhá vždy bezprostředně před Huffmanovým kódováním. V implementaci je RLE aplikováno vždy po využití modelu. Pokud tedy data nejsou předzpracována modelem, RLE použito není. Tohle výchozí chování lze upravit změnou hodnoty řídící proměnné `use_rle` ve funkci `main`.

2.2 Model

Jako model umožňující zvýšit kompresní účinnost je využita diference sousedních hodnot (pixelů). První hodnota je ponechána a každá z ostatních je transformována na rozdíl mezi svojí hodnotou a hodnotou předcházející, tedy dle vzorce

$$y_n = \begin{cases} x_n & n = 1 \\ (x_n - x_{n-1}) \mod 256 & n > 1 \end{cases}.$$

2.3 Huffmanovo kódování

Pro implementaci byla zvolena kanonická verze Huffmanova kódování. Kódová slova pro jednotlivé symboly (pixely 0-255) jsou spočteny algoritmem z přednášek. Konec kódovaných dat je vždy označen kódovým slovem pro speciální symbol konce dat s hodnotou 256. Program podporuje kódová slova o délce nejvíce 64 bitů. Pro

delší kódová slova není chování programu korektní. Před samotná zakódovaná data je vždy vložena kódová kniha, která je reprezentována jako uspořádaná trojice, jejíž první prvek nese délku nejdelší značky, následuje pole počtů symbolů o jednotlivých délkách od délky 1 do uvedené největší délky, za kterým se nachází pole jednotlivých kódovaných symbolů uspořádaných vzestupně dle délky a hodnoty kódového slova. Speciální symbol konce dat v kódové knize uveden není. Dekódování je pak realizováno pomocí polí `first_code` a `first_symbol` opět způsobem uvedeným v přednáškách.

2.4 Skenování obrazu a komprese

Program poskytuje dvě varianty skenování dat (obrazu) – sekvenční a adaptivní. V případě sekvenčního skenování je obraz procházen horizontálně, přičemž je celý chápán jako jeden souvislý blok dat. Adaptivní skenování oproti tomu rozkládá obraz na menší bloky dat o pevné velikosti $32 \times 32 = 1024$ pixelů (bytů), přičemž pro každý z nich je volen horizontální, či vertikální průchod tak, aby bylo dosaženo co největší kompresní účinnosti. Typ průchodu je u adaptivního skenování pro každý blok uložen v bytu který bezprostředně předchází jeho komprimovaná data, aby byla tato informace zachována pro dekompresi bloku.

U obou variant skenování obrazu je blok dat v případě, že pro něj nelze dosáhnout komprese, uložen do výsledných komprimovaných dat nekomprimovaně. To, jakým způsobem jsou data bloku uložena ve zkomprimovaných datech (komprimovaně, či nekomprimovaně), udává hodnota prvního bytu komprimovaného bloku.

V rámci komprese je každý blok dat zpracován zcela samostatně. Pro každý blok je tedy zkonstruován nový Huffmanův kód na základě jeho obsahu, přičemž ani případné předzpracování dat pomocí modelu či RLE neprobíhá napříč bloky, nýbrž vždy pro každý blok zvlášť.

3 Implementace a překlad

Program je implementován v jazyce C++20[1]. Zdrojové soubory jsou překládány pomocí GCC[2] (`g++`). S využitím nástroje GNU_{make}[3] lze zdrojové soubory přeložit příkazem `make`, který na základě přiloženého Makefile souboru vytvoří spustitelný soubor `huff_codec`.

4 Spuštění a ovládání

Program lze spustit příkazem

```
./huff_codec [-c|-d] [-m] [-a] -i <ifile> -o <ofile> [-w <width_value>] [-h],
```

kde

- `-c` – Vstupní soubor je komprimován (výchozí režim činnosti aplikace).
- `-d` – Vstupní soubor je dekomprimován.
- `-m` – Aktivuje model a RLE při zpracování vstupních dat (parametr zadáván v případě komprese i dekomprese).
- `-a` – Aktivuje režim adaptivního skenování obrazu, jinak je použito sekvenční skenování v horizontálním směru bez dělení na bloky (parametr zadáván v případě komprese i dekomprese).
- `-i <ifile>` – Název vstupního souboru (dle režimu činnosti aplikace buď data určená ke kompresi, nebo dekompresi)
- `-o <ofile>` – Název výstupního souboru (dle režimu činnosti aplikace buď komprimovaná, nebo dekomprimovaná data).
- `-w width_value` – Udává šířku obrazu, přičemž `width_value` ≥ 1 (parametr zadáván pouze v případě komprese, aby program věděl, jak široký obraz je).
- `-h` – Na standardní výstup je vypsána nápověda a program je ukončen.

5 Vyhodnocení efektivity

V rámci testování funkčnosti aplikace bylo vyhotoveno její vyhodnocení efektivity na dostupných testovacích datech. Vyhodnocení bylo provedeno na referenčním serveru **merlin.fit.vutbr.cz**. Následující tabulky 1 a 2 uvádějí naměřené výsledky.

[1] Dokumentace dostupná z https://en.cppreference.com/w/
[2] Dostupné z https://gcc.gnu.org/
[3] Dokumentace dostupná z https://www.gnu.org/software/make/manual/make.html

obrazová data	entropie	vstupní parametry							
		–		`-m`		`-a -w 512`		`-a -w 512 -m`	
		bpp	ct [s]	bpp	ct [s]	bpp	ct [s]	bpp	ct [s]
df1h.raw	8.00000	8.00003	0.011	0.02371	0.004	5.35986	0.008	0.37939	0.008
df1hvx.raw	4.51396	4.58450	0.010	0.95977	0.007	3.77783	0.008	1.09658	0.011
df1v.raw	8.00000	8.00003	0.011	0.03353	0.004	5.35986	0.011	0.38134	0.007
hd01.raw	3.82550	3.88021	0.010	2.68310	0.007	4.01791	0.016	3.09005	0.022
hd02.raw	3.63593	3.70529	0.012	2.63803	0.007	3.93685	0.014	3.06115	0.022
hd07.raw	5.57693	5.61303	0.010	3.32278	0.008	4.70617	0.013	3.29205	0.020
hd08.raw	4.21081	4.23916	0.009	2.97671	0.011	3.94143	0.016	3.00903	0.021
hd09.raw	6.62114	6.66030	0.011	4.61242	0.011	5.97262	0.016	4.50048	0.025
hd12.raw	6.16570	6.20101	0.011	3.84118	0.008	5.33410	0.017	3.80169	0.024
nk01.raw	6.47290	6.50881	0.012	6.04809	0.014	6.55416	0.015	6.03460	0.028
průměr	**5.70228**	**5.73924**	**0.010**	**2.71393**	**0.008**	**4.89608**	**0.013**	**2.86464**	**0.018**

Tabulka 1: Entropie jednotlivých datových zdrojů, počty bitů na znak (bpp) při jejich kompresi a doba této komprese (ct) v sekundách (s) v jednotlivých režimech činnosti při kompresi (sekvenční skenování bez modelu, sekvenční skenování s modelem, adaptnivní skenování bez modelu, adaptivní skenování s modelem) pro RLE aplikované pouze v případě využití modelu.

V tabulce 1 lze pozorovat, že nejlepšího kompresního poměru dosahuje pro 6 z 10 testovacích zdrojů sekvenční skenování s využitím modelu, pro zbývající pak adaptivní skenování s využitím modelu. Výsledky adaptivního skenování jsou ovlivněny zvolenou velikostí bloku, mohlo by je tedy případně jít lehce zlepšit nalezením jeho vhodnější velikosti (výchozí velikost hrany bloku je 32, velikost bloku je tedy $32 \times 32 = 1024$). U zdrojů df1h.raw a df1v.raw si povšimněme, že v případě sekvenčního skenování bez využití modelu ke kompresi nedochází vůbec. Při povolení RLE bez využití modelu pak nelze dosáhnout komprese při sekvenčním skenování bez využití modelu pouze u zdroje df1h.raw, jak vyplývá z tabulky 2. Rovněž lze pozorovat, že povolení RLE i bez využití modelu vede ke zlepšení kompresního poměru, nikoli však k tak signifikantnímu, jako s využitím modelu i RLE současně.

obrazová data	vstupní parametry			
	–		`-a -w 512`	
	bpp	ct [s]	bpp	ct [s]
df1h.raw	8.00003	0.012	0.67626	0.013
df1hvx.raw	2.41967	0.006	1.62747	0.012
df1v.raw	0.05874	0.004	0.67626	0.011
hd01.raw	3.03042	0.006	3.44512	0.023
hd02.raw	2.88067	0.006	3.35208	0.024
hd07.raw	4.67449	0.009	4.34356	0.022
hd08.raw	3.38995	0.007	3.49011	0.019
hd09.raw	6.43814	0.010	5.93515	0.030
hd12.raw	5.29965	0.009	5.00991	0.027
nk01.raw	6.47579	0.011	6.55050	0.027
průměr	**4.26675**	**0.008**	**3.51064**	**0.020**

Tabulka 2: Počty bitů na znak (bpp) a čas trvání komprese (ct) v sekundách (s) pro jednotlivé datové zdroje bez využití modelu při kompresi, avšak s aplikovaným RLE pro sekvenční i adaptivní variantu skenování.